6-2012

Distributed Incomplete Pattern Matching via a NovelWeighted Bloom Filter

Siyuan LIU
Carnegie Mellon University

Lei KANG
Hong Kong University of Science and Technology

Lei CHEN
Hong Kong University of Science and Technology

Lionel NI
Hong Kong University of Science and Technology

DOI: https://doi.org/10.1109/ICDCS.2012.24

Distributed Incomplete Pattern Matching via a Novel Weighted Bloom Filter

Siyuan Liu *, Lei Kang #, Lei Chen #, Lionel Ni #

*iLab, Heinz College, Carnegie Mellon University
#Hong Kong University of Science and Technology

***Abstract*—In this paper, we first propose a very interesting and practical problem, pattern matching in a distributed mobile environment. Pattern matching is a well-known problem and extensive research has been conducted for performing effective and efficient search. However, previous proposed approaches assume that data are centrally stored, which is not the case in a mobile environment (e.g., mobile phone networks), where one person's pattern could be separately stored in a number of different stations, and such a local pattern is incomplete compared with the global pattern. A simple solution to pattern matching over a mobile environment is to collect all the data distributed in base stations to a data center and conduct pattern matching at the data center afterwards. Clearly, such a simple solution will raise huge amount of communication traffic, which could cause the communication bottleneck brought by the limited wireless bandwidth to be even worse. Therefore, a communication efficient and search effective solution is necessary. In our work, we present a novel solution which is based on our well-designed *Weighted Bloom Filter* (WBF), called, *D*istributed *I*ncomplete pattern *matching* (*DI-matching*), to find target patterns over a distributed mobile environment. Specifically, to save communication cost and ensure pattern matching in distributed incomplete patterns, we use WBF to encode a query pattern and disseminate the encoded data to each base station. Each base station conducts a local pattern search according to the received WBF. Only qualified IDs and corresponding weights in each base station are sent to the data center for aggregation and verification. Through extensive empirical experiments on a real city-scale mobile networks data set, we demonstrate the effectiveness and efficiency of our proposed solutions.**

***Keywords*-Incomplete pattern matching, time series, distributed mobile environment, Weighted Bloom Filter.**

I. Introduction

Given a pattern time series, finding time series in database that are *similar* to the given pattern (a.k.a. *pattern matching*) is a well-studied problem due to its wide application domain. Most existing solutions assume that data are centrally stored[2], [13], [16], [1], [3], [6], in other words, all the time series data are collected to a data center. Under this assumption, pattern matching is conducted locally over the collected time series. However, in a distributed wireless mobile environment, a person's mobile phone communication data are often distributed in several base stations. Even just for one day, his or her communication data may be recorded in different base stations (e.g., near office or home) at different time. Since the data stored in each base station are not complete compared to data collected in the data center, we call such distributed data as *incomplete* data and define a new problem, *Incomplete Pattern Matching* (IPM), that is, finding the pattern over incomplete data in a distributed mobile environment. Our focus targets on retrieving the pattern of interest in a communication and time efficient way. The pattern that we are interested is the communication data (number of calls, call duration and number of partners, etc.) of a person along the time. The existing solutions for centralized pattern matching are not suitable for incomplete data unless all the data are collected at the data center. However, such a solution incurs huge amount of communication cost, which might degrade the search performance onwards.

In fact, solutions for pattern matching over incomplete data can be applied in many applications. For instance, a mobile phone networks service provider wants to promote a certain call package service to customers. Usually the customers having similar communication patterns would prefer similar call package services, thus given a preferred customer's pattern, the service provider needs to run a pattern matching query over the networks to find customers having the similar patterns. Though it is useful, it is quite challenging to perform pattern matching over distributed incomplete data.

Challenge 1 (Incomplete collecting): The data of a particular person is usually stored in several local base stations. Usually a base station may serve numerous mobile phones in one cell, and a data center may serve vast base stations. The communication data of a person could be distributed in a number of base stations. We call the communication data stored in several base stations along a person's moving trajectory as a *global pattern*, and accordingly the communication data stored at each base station as a *local pattern*. The distributed mobile environment causes the pattern data to be incompletely stored in different base stations, and the distributed incomplete data would cause either inaccurate pattern matching results or high communication and storage costs.

Challenge 2 (Dynamic matching): Global and local patterns change along the time, which brings great challenges to conduct pattern matching with high accuracy and efficiency.

To tackle the above major challenges, we propose *DI-matching*, a framework for efficiently and accurately retrieving patterns from distributed incomplete data sets. The

whole process consists of three steps. First, we represent and encode the given patterns utilizing our novel *Weighted Bloom Filter* (WBF), where the weight encodes the relation of local patterns and a corresponding global pattern along the time. Second, the represented patterns stored in each base station could be sampled and hashed into WBF to check whether it is the pattern of interest. Matched patterns together with the corresponding weights are submitted to data center. Third, the data center aggregates the weights of the patterns and ranks them by weight values.

Our contributions are summarized into three-folds. First, we define *communication pattern* time series based on practical applications, and propose a new problem, *Incomplete Pattern Matching* (IPM), to study the pattern matching in terms of incomplete, dynamic and distributed data in a distributed environment. Second, we design a novel *Weighted Bloom Filter* (WBF), based on which, we devise a communication-efficient and search-effective framework, called *DI-matching*, to address IPM problem. Third, we evaluate our methods with extensive empirical experiments. The results confirm the efficiency and effectiveness of the proposed solutions.

The rest of the paper is arranged as follows. The preliminary information of our study is provided in Section II. Section III defines IPM problem. In Section IV, we devise *DI-matching* based on WBFs. The empirical experiments to evaluate the merits of our methods are elaborated in Section V. Section VI gives an overview of related work. We conclude and outline our future work in Section VII.

II. PRELIMINARY

First, we introduce the mobile phone communication data sets used in our study. Second, we define *communication pattern* time series. Third, we briefly explain Bloom Filters.

A. Mobile Phone Communication Data Sets Basics

We were able to collect mobile phone data from a city in China from January 1^{st}, 2008 to December 31^{st}, 2008. The city area is around 8700 km^2 covered by 5120 base stations, serving about 3.6 million mobile phone users. Some raw data characteristics are listed as follows.

Characteristic 1 (Distributed data sets): In the mobile phone networks, the communication (call) data are distributively stored in base stations, and usually uploaded to data center once a month. A person's communication data are always stored in several base stations. Such a data characteristic results in a distributed incomplete data collection, and we cannot aggregate all the data to conduct the pattern matching due to high communication cost. Hence an accurate and efficient distributed pattern matching method is required to handle the incomplete pattern data.

Characteristic 2 (Dynamic evolving data): In the mobile phone networks, the communication data are generated when a call is built up. That is the communication data are evolving along the time. Such data characteristic requires that the proposed pattern matching method can compute and update the results efficiently.

Table I
MEANING OF SELECTED NOTATIONS IN THE PAPER

Variables	Description
g	Time interval
T	Time series, $g \in T$
γ_i^g	Pattern in time interval g
Γ^g	The pattern set in the time interval g
ε	User-specified approximation parameter
O_i	Data object (mobile phone)
Z	Universe of data objects
N_0	Data center node
N_j	Base station node
V_i	Global value of data object O_i
$V_{i,j}$	Local value of O_i at node N_j
a	Number of patterns
b	Number of samples in the pattern
k	Number of hash functions
m	Length of Weighted Bloom Filter
p	Probability of a bit to be 0 in WBF
q	Probability of a value to be in WBF

Characteristic 3 (Large scale data sets): In our collected data sets, there are 3.6 million mobile phones, and the raw data per day are more than 2 GB with more than 10 million records. Hence, the search and storage costs over these data are extremely expensive.

In the mobile phone communication data sets, there are mainly three attributes for a person's communication. *Number of calls* record the call count that a person makes within a time interval. *Call duration* is the total duration of the calls a person makes within a time interval. *Number of partners* is the number of distinct persons that a person contacts within a time interval. Hence we take these three attributes as an example to define the *communication pattern* of a person in a mobile phone communication networks. For convenience, Table I summarizes the notations used in this paper. We use person, mobile phone and object interchangeably to express the same concept.

In practice, the communication companies would like to take the communication pattern of a person as an overall combination of the number of calls, call duration, and number of partners, etc., and the formal definition is as follows.

Definition 1: Given a person O_i in mobile phone networks, a time interval g, O_i has the attribute set $S_i^g = \{s_i^{g,1}, ..., s_i^{g,f}, ..., s_i^{g,m}\}$, where $1 \leq f \leq m$ and m is the number of attributes. The communication pattern within the time interval g, γ_i^g is

$$\gamma_i^g = \frac{1}{m} \sum_{f=1}^{m} w_f s_i^{g,f} \tag{1}$$

where w_f^g is the weight of an attribute f in the time interval g. The communication pattern set is $\Gamma^g =$

$\{\gamma_1^g, ..., \gamma_i^g, ..., \gamma_n^g\}$, where $i \in \mathbb{N}$, $i = 1, 2, ..., n$, and n is the number of persons.

In our study, we take the mean of three attributes (m=3), the number of calls, the call duration, and the number of partners, as pattern values. Our solution can be extended to other cases where $m > 3$ and other functions. The default time interval is set as a minute, which is set by the mobile service company. The time interval does impact the matching results. Hence, in our paper, we hold the time interval as a minute. In fact our method can work in different time intervals via simple adjustments. We say a pattern matches with the given one if and only if their global patterns match.

B. Bloom Filter Basics

Bloom Filter (BF) [7] is a simple space-efficient randomized data structure for approximately representing a set to support membership queries. BF allows false positives, but the space savings often outweigh this disadvantage when the probability of the error is sufficiently low. Burton Bloom introduced Bloom Filter in 1970 [7], and ever since it has been widely used in database and networking applications [8]. Bloom Filter can be considered as an array of 0s and 1s. When a value is hashed into the Bloom Filter, the corresponding bits are turned to 1 from 0. To check whether a value is in a given set, we can hash the value into the Bloom Filter to check whether all the corresponding bits are 1s. It allows a false positive, and gurranttee no false negative. BF has significant space and time advantages [7], [8], but one notable issue in Bloom Filter is that the accuracy may not be good enough, that is, it only guarantees a false positive probability lower bound [21], [22]. In our study, we design *Weighted Bloom Filter*, to address the incomplete pattern matching problem. The major difference between WBF and BF is that each bit with 1 in WBF has a pointer pointing to the weight of corresponding hashed values. With this scheme, the false positive probability could be significantly reduced.

III. Problem: Incomplete Pattern Matching

In this section, we define a new pattern matching problem, *Incomplete Pattern Matching*, which is different from traditional pattern matching methods in two aspects. First, patterns are incomplete. Second, the environment is distributed in a large scale.

A. Running Example

In the following parts, we will refer to the example scenario that represents a typical application of incomplete pattern matching. It will serve to motivate our approach and we will use it to validate our incomplete pattern matching algorithms. The example application we consider is to provide call package service to mobile phone customers, which requires to search similar communication patterns across a distributed set of base stations. Numerous service providers are thirsting for retrieving their target customers according to their pattern examples (given a user pattern). Following is a continuous searching query:

Searching Query. Which mobile phone users whose communication patterns are the most similar ones to the given pattern across all base stations?

Obviously, along the time the number of service providers, patterns, and base stations are accumulating and evolving. This query is required to provide near real time feedback to service providers (in the data center), allowing them to adapt their advertising strategies and budgets in response to observed similar patterns. Furthermore, the effects of changes made to the data center would be observable shortly afterward, achieving tight closed-loop interaction with immediate feedbacks. Thus, online realtime monitoring is preferable to off-line analysis in this scenario.

In the next subsection, we formally define the *Incomplete Pattern Matching* problem.

B. Problem Definition

We consider a distributed online searching environment with $l + 1$ nodes: a data center node N_0, and l remote base station nodes N_1, N_2, ..., N_l. Collectively, the base station nodes monitor a set Z of *n* objects (mobile phones) $\mathrm{Z} = \{O_1, O_2, ..., O_n\}$, each object O_i is associated with a set of natural numbers V_i, where $\mathrm{V}_i = \left\{\gamma_i^1, \gamma_i^2 ..., \gamma_i^t, ...\right\}$ (each represents a value of its corresponding attribute, and t is a time interval). The complete global values of an object may not be observed by an individual node (base station). For an individual base station node N_j, the partial (local) values of object O_i are denoted as $\mathrm{V}_{i,1}, \mathrm{V}_{i,2}, ..., \mathrm{V}_{i,j}$. The overall (global) values of object O_i, which is not materialized on any node, is defined to be $V_i = \sum_{1 \le j \le l} V_{i,j}$.

We define the *Incomplete Pattern Matching* (IPM) problem as follows.

Problem Statement: Incomplete Pattern Matching (a top-K query)

Given a user-interested time series pattern V_u and a user-specified approximation parameter ε, find a set Ω of the time series V_i such that

$$\forall \gamma_u^t \in V_u, \forall \gamma_i^t \in V_i, \left|\gamma_u^t - \gamma_i^t\right| \le \varepsilon \tag{2}$$

where $\varepsilon > 0$, and the number of returned results through the above query is K, where K is a number indicating the top-K most similar patterns ($|\Omega| = K$). □

If $\varepsilon = 0$, the data center reports the exact top-K set. For non-zero values of ε, a corresponding degree of errors are tolerant in the reported top-K set. The objective of retrieving top-K similar object patterns in the incomplete patterns is to provide, in the data center, an approximate top-K set that is valid within ε over the time, while *minimizing* the *costs* in communication and storage.

In this paper, we take the similarity distance function defined in Eq. (2) as an example. Because in the mobile

phone networks, the communication data are computed at every a time interval, and we call two persons having a similar communication pattern if and only if they are similar in each time interval. Hence we utilize L_1-norm similarity distance function. At the same time, the mobile phone networks data (the number of calls, the call duration, and the number of partners) are all integers, hence we discuss the integer case in our work. For the decimal numbers or other distance functions, we leave them as future work.

C. Analysis and Observation

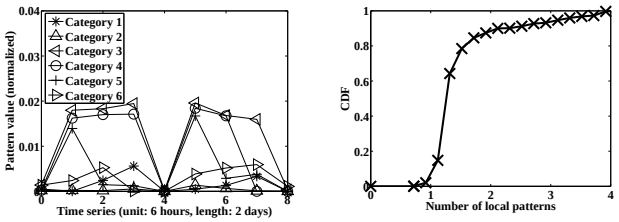
The challenge here is that a person's data are stored in several base stations, say he or she lives in one place but works in another. We call each piece of data stored in each base station is a local pattern, and the gathering of the whole data is the global pattern.

Intuitively, we may have two approaches to conduct the pattern matching task in the distributed mobile environment.

Approach 1: we may ship the global data from all the nodes to the data center, which conducts their aggregation and then applies the pattern matching algorithms in time series. This approach is clearly inefficient and often infeasible due to the communication and storage costs between the data center and nodes, when there are huge number of nodes with a large amount of time series data.

Approach 2: Each base station locally employs pattern matching algorithms on its own data set to find the similar ones, and the data center aggregates the local matched patterns to find the global ones. This approach reduces communication overhead, but may miss items which are unmatched in base stations, while their aggregated patterns across all nodes match with the given one. On the other hand, even a pattern in a base station matches with the given one, we cannot ensure this one is a "real" target. For instance, assume that a given pattern is $\{3, 4, 5\}$, where there are 3 base stations respectively having three patterns $\{1, 1, 1\}$, $\{2, 2, 0\}$, and $\{0, 1, 4\}$. Obviously, the three patterns are individually unmatched with the given one, but when we aggregate the three patterns, we can find the global pattern exactly matches with the given one. However, given the same pattern, and the 3 base stations respectively posses three patterns $\{3, 4, 5\}$, $\{3, 4, 5\}$, and $\{3, 4, 5\}$. We can find even if the individual patterns exactly match with the given pattern, the global one $\{9, 12, 15\}$ is totally different from the given one.

Nevertheless, the data center relies on data or communications (messages) from distributed nodes to retrieve the target patterns. The above naïve solutions explained that messages that contain the global data sets are inefficient, and messages that contain only the local matched patterns are lossy. The main reason is that the local data are incomplete comparing to the global data. Thus, the core problem of incomplete pattern matching is to devise an appropriate scheme for summarizing given patterns and local data in time series. In other words, each node should be able to feedback *small yet informative* messages to the data center so that the center can derive global matched patterns in an efficient and accurate fashion.



(a) Periodicity, divisibility (b) Similarity

Figure 1. Realistic pattern features

Before presenting our novel solution, we introduce a number of interesting discoveries from the raw data sets.

Observation 1: In the global perspective, the patterns appear periodical and divisible features. In Figure 1 (a) (pattern values are normalized to the mean value), we show the communication patterns of six population categories (based on occupations) in two days. Based on our data sets, we can observe that in each day, the pattern shapes are similar. The periodical data are helpful for us to speed-up computation and save storage. In our solution, we utilize the time order in the period to conduct the matching. We *accumulate* the values in the original pattern along the time, and find that the patterns in different categories are more divisible over the time.

Observation 2: In the individual perspective, the local patterns of similar global patterns appear some interesting similarity. In Figure 1 (b), we show the local pattern similarity (as defined in Eq. (2)) distribution of the similar global patterns. X-axis is the number of local patterns which are similar, and y-axis is the percentage in Cumulative Distribution Function (CDF). We can observe that in the similar global patterns the percentage that there exist more than one similar local patterns is greater than 90 %. In other words, within the same communication category, even the local patterns are variational, at least one local pattern is similar. The similar observations could be found in [15], [10], [14], [11], [12]. It is easy to understand that in our real life, we say two persons in the same category (similar communication patterns), and then they may have similar daily behaviors (e.g., occupations, routes, and schedules). We utilize this observation to design a Weighted Bloom Filter and conduct pattern matching in a distributed mobile environment.

IV. Framework: DI-matching

The overview of *DI-matching* is explained in Figure 2. First, a set of given patterns (original patterns, e.g. a global pattern with corresponding local patterns) are represented and encoded at data center side. Second, a Weighted Bloom Filter is constructed on the represented and encoded patterns.

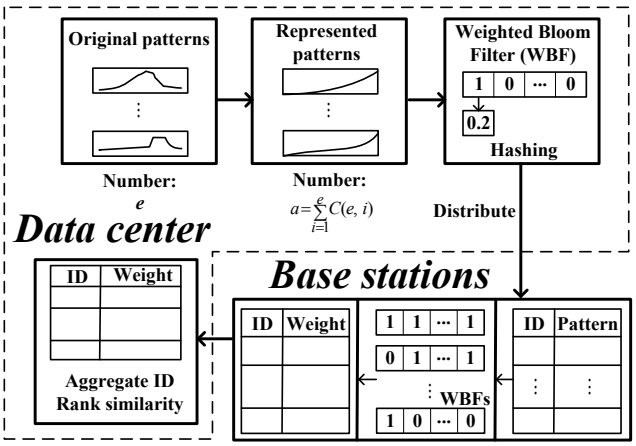


Figure 2. Distributed incomplete pattern matching overview

Third, the Weighted Bloom Filter is distributed to base stations where the pattern matching is conducted. Finally, the similar patterns' IDs and weights are sent back to data center to decide the top-K similar patterns.

A. Pattern Representation and Encoding

Bloom Filter is designed to tell whether a value we want exists or not, but cannot distinguish a value sequence. For example, a Bloom Filter may consider $\{1, 2, 3\}$ and $\{3, 2, 1\}$ as the same pattern because the values 1, 2 and 3 are the same in the two patterns, but in our application scenario, $\{1, 2, 3\}$ and $\{3, 2, 1\}$ are two totally different time series patterns. Hence to distinguish the patterns with same values, we develop a pattern representation and encoding method which utilizes the time information hidden in the pattern values.

Recalling Definition 1 to define a time series pattern, that is a communication pattern (pattern, for short in the following parts), e.g. $\{1, 2, 3\}$, contains two pieces of information, the value at each time interval and the order of this value along the time. We utilize these information together to represent and encode the pattern as follows.

Given an original pattern V^T, for its value at time interval g, V^g, we transform it into its accumulation form $f(g)$, where

$$f(g) = f(g-1) + V^g \tag{3}$$

$f(0) = V^0$, and $g = 1, ..., t$.

In other words, the value $f(g)$ of the new form is the accumulated value of the original pattern. For example, $\{1, 2, 3\}$ is the original pattern and the new one turns to be $\{1, 3, 6\}$. We transform the original pattern to this accumulation form for three merits. First, accumulation form can represent the patterns with same values into a unified monotonic form, which is more straightforward and easy to distinguish the actual different patterns. Second, the difference between different patterns may be enlarged at each time point, because the difference is also accumulated as the accumulation of the value itself. Third, the attribute value and the order information are combined into the new value in its accumulation form. Given a value of the new form, we can easily estimate the original value and the time interval as long as the scale and tail values are known. Thus, we can make comparisons among patterns, regardless the orders in them (they are already ordered in the accumulation form). In our application scenario, the pattern values are time series, which means the sequence is important. The accumulated calling patterns can represent a person's calling behavior much better than the calling patterns in individual time intervals because the accumulation method is able to enlarge the pattern difference and the sequence difference in time series.

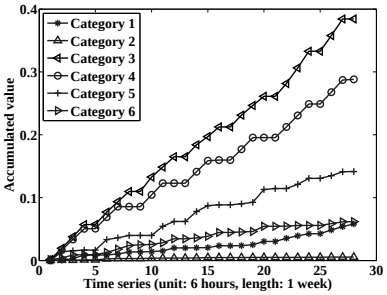


Figure 3. Pattern representation

For example, the time series patterns $\{1, 2, 3\}$ and $\{3, 2, 1\}$ (the value sequence in the vector is the time sequence) can be transformed as $\{1, 3, 6\}$ and $\{3, 5, 6\}$, which can be easily distinguished as the values are different at 1 and 5 (the first two values), and the values at the same time indicate the order of the original values (5 is latter than 3 because 5 is larger than 3).

We illustrate the results of the method by real life data in Figure 1 (c). X-axis is the time series, with the unit being six hours and the time length being two weeks. We accumulate the values in the original pattern by Eq. (3). We can observe that the accumulated patterns are not only distinguishable by values but also embedded the time information.

Suppose we receive a pattern set, which consists of three local patterns, e.g. one user's home, work place and shopping place are in different base stations. Another user's home and work place are in the same base station, hence we cannot match the pattern with the user's pattern with a single pattern or global pattern comparison. In the base station, we need to compare the pattern of each person with all the possible combinations of the given patterns. Given a pattern set consisting of one global pattern and several local patterns, we need to compare the pattern of each person with each pattern in the pattern set. Say the patterns in the pattern set are collected from one person traveling l base stations, the

Algorithm 1 *Data Center Side*: Weighted Bloom Filter

Input: local pattern set S, number of local patterns $e(e = |S|)$ and sample number b
Output: Weighted Bloom Filter set WBF

Begin
represent patterns by Eq. (3), get S'
take the combinations of pattern S'
get the new pattern set S'' and $a = |S''|$
for pattern i **do**
  sample b points ($v_{ij}, i = 1, 2, ..., a$ and $j = 1, 2, ..., b$) and the maximum point is v_{ib}
  for each point **do**
    the weight is $w_{ij} = \frac{v_{ib}}{v_{ab}}$
    hash each value v_{ib} into WBF using k hash functions, and add the weight of this point to the pointer of the bits that turned to be 1s.
  end for
end for
return WBF.
End

Algorithm 2 *Base Station Side*: Pattern Matching

Input: Weighted Bloom Filter and the pattern set
Output: Weights of the patterns or zero

Begin
convert a pattern into an accumulated form and sample b points;
for each point $v_i(i = 1, 2, ..., b)$ **do**
  hash into WBF
  if corresponding bits contain 0 **then**
    return zero
    if all bits contain the same weight **then**
      put the weight into an array
    end if
  end if
end for
if all the weights of every point are the same **then**
  return the weight
  else return zero
end if
End

number of comparison Ψ is

$$\Psi = \sum_{j=1}^{l} C(l, j) \tag{4}$$

where $C(l, j)$ is the number of j-combinations from a given set of l elements.

Obviously there exists significant time cost in this procedure if we simply adopt traditional pattern matching methods. Thus we hash all the patterns into one Bloom Filter and then distribute this Bloom Filter to all the base stations. In each base station, we just hash each person's converted pattern into this Bloom Filter to see if this pattern belongs to the given pattern set. If the target bits are all 1s, then we say this person is a target person, otherwise no. Weighted Bloom Filter is designed in the following subsection.

B. Weighted Bloom Filter

Weighted Bloom Filter is an extended version of Bloom Filter, where each bit equals to 1 points to a weight value, which can reduce and upper-bound the false positive probability.

Suppose the given global pattern is $\{3, 4, 5\}$, with two local patterns $\{1, 2, 3\}$ and $\{2, 2, 2\}$. There are two persons in the base station, with the patterns, say $\{3, 4, 5\}$ and $\{1, 2, 3\}$. They are both possible target person but in different levels (global-matched or local-matched). Unfortunately, a Bloom Filter cannot distinguish the two patterns because Bloom Filter only decides whether a value exists. Hence, we devise a novel *Weighted Bloom Filter* (WBF) to distinguish the global-matched ones and local-matched ones, and then inform the data center the similarity (values) of the selected persons. The center combines the value of the same person from different base stations, ranks the similarity, and delivers the final results to the users.

The advantages of WBF are three-folds. First, the communication cost between base stations and data center is highly reduced, because we can only send the ID and weight back to data center, instead of ID and time series pattern. Second, we can learn the similarity in a much more precise style. The pattern matching in base stations can distinguish the global-matched or local-matched patterns. Third, the false positive rate is reduced by the calculation of weights, because only all the values of a pattern having the same weight is accepted. For example, there are two patterns in Bloom Filter $\{1, 2, 3\}$ and $\{2, 4, 5\}$, a pattern $\{1, 4, 5\}$ (an unmatched pattern) returns a matched result in Bloom Filter, but in WBF, $\{1, 4, 5\}$ returns an unmatched result so that it will not be accepted as a matched result.

In a WBF, each pattern and each value of each pattern are attached with weights. The principles of WBF are described as below.

- If a pattern of a person matches with a given global pattern, the weight of this pattern in WBF is 1.
- If a pattern of a person matches with a local pattern of the given global pattern, and the weights of all local patterns from all the base stations belonging to this person is summed up to 1, if this person is a "real" target person.
- The weight of each value in a pattern is equal to the weight of this pattern.

The WBF design details are introduced as follows.

First, the pattern representation and weight assignment are conducted in the data center. Suppose we get a pattern

Algorithm 3 *Data Center Side*: Similarity Ranking

Input: person IDs and the corresponding weights (suppose r pair totally)

Output: the top-K target persons

1: **Begin**
2: **for** each ID **do**
3: take the sum of the weights, if larger than 1, delete this ID
4: rank the IDs in descending order
5: take the top-K IDs and return them
6: **end for**
7: **End**

from a user that consists of several local patterns, the sum of which is a global pattern. Note that if and only if the global patterns match, we call a target pattern. We convert the given patterns to the accumulated forms and calculate all possible combinations of these new patterns (there are a new patterns, $a > 1$, and in the real life applications, a is always a large number). The weight of a local pattern is the maximum value in it over the maximum value in the global pattern, e.g., the weight of a pattern $\{1, 2, \mathbf{3}\}$ is 3/9, with respect to the global pattern $\{4, 7, \mathbf{9}\}$. All the weights of the patterns are maintained by an array W.

To further save the communication, storage and computation costs, we can utilize a sampling scheme on patterns. For each of these patterns (the number of these patterns is a), we will sample b values (uniform sampling) and set the weight of each value as the weight of this pattern. There are totally $n = ab$ values that should be hashed into a WBF with length m. This filter begins as an array of all 0s. Each value is hashed k times by different hash functions, and each hash yields a bit location which is set to 1. If one bit is set to 1 from 0 in WBF by a value, this bit is added into a queue that contains the weight of this value. If this bit is already set to 1, the weight of this value is pushed back to the queue. Note that in our problem definition, we define the approximate similarity, hence when we conduct the hashing, we hash all the possible approximate values into WBF. We summarize this method in Algorithm 1, which constructs WBFs of given patterns in the data center. First, we represent the given patterns by Eq. (3). Second, the sample method is provided to compress the represented patterns. Third, we calculate the weights of patterns. At last, the given patterns are hashed into a WBF.

Second, the pattern matching is carried out in base stations. In base stations, the patterns to be checked are processed by the same above methods. The sampled values of a pattern is hashed into WBF. If all the target bits are 1s and all the weights are the same (1 bit may contain 2 or more weight value), this pattern is returned as a target pattern. The number of pattern weight calculations is bounded by nk, as each bit of 1 contains a weight at most and the sampled b points are hashed into bk bits at most. We summarize this method in Algorithm 2, which carries out the pattern matching in base stations and reports possible matched patterns to the data center.

Algorithm 2 is carried out in base stations to conduct the pattern matching. First, the patterns in the base stations are represented and sampled by the same methods we discussed in Algorithm 1. Second, we match the values of the patterns in the Weighted Bloom Filter from the data center, and if and only if the weights of every value (point) are the same, we now find the possible matched patterns. At last, the possible matched ones are sent back to the data center with the IDs and weights.

Last, the matched patterns are submitted to the data center, and similarity ranking is provided. The data center adds up all the pattern weights of the same person, deletes the sum of the weights larger than 1 and ranks the persons in descending order from 1. The top-K persons are finally returned. The reason for the weight sum larger than 1 is that a person's local patterns in different base stations may match with the given global pattern and also match with the given local patterns. Obviously this person is not our target person because this person's aggregated global pattern is different from the given global pattern. The method is summarized in Algorithm 3, which ranks the similarities of the patterns by the weights.

The time complexity of the three algorithm are analyzed as follows.

Algorithm 1 time complexity: We need to hash $n = ab$ values into Bloom Filter by k hash functions, so the time complexity is $O(nk)$.

Algorithm 2 time complexity: We need to hash b values into WBF by k hash functions, and the time complexity is $O(bk)$. The comparison of weights costs kb at most. Totally, the time complexity is $O(bk)$.

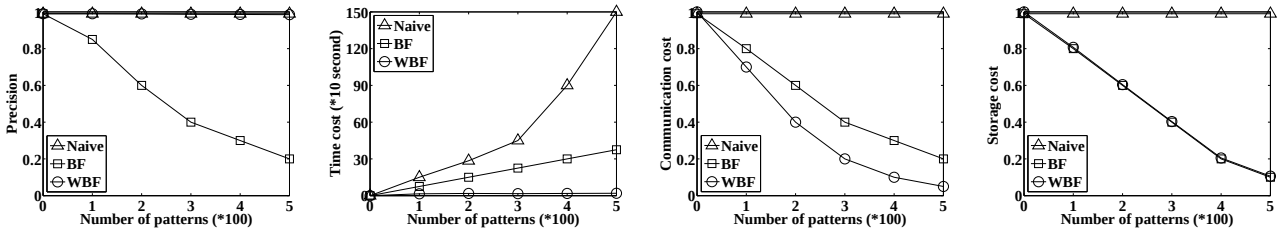
Algorithm 3 time complexity: We travel all the IDs and cost $O(r)$, while the ranking costs $O(r \log r)$, thus, totally the time complexity is $O(r \log r)$.

V. EMPIRICAL EVALUATION

In this section, we evaluate our proposed methods using large scale real life data. First, we give the experiment description including real life data sets, experiment setup and comparison methods. Second, we investigate the parameter setting impacts in our method. Base on the parameter study, we conduct the experiments to evaluate accuracy, efficiency and effectiveness of our solutions and comparison methods. At last, we demonstrate the upper bound tightness of WBF by the empirical experiment results.

A. Experiment Description

Data sets description: The data set scale is around 1 Tera-Bytes. The data we collected from each base station, are called *Call Detail Record* (CDR) and *Cell Detail List*



(a) Accuracy (b) Time cost (c) Communication (d) Storage cost

Figure 4. Accuracy and efficiency evaluation

(CDL). Basically, CDR contains mobile phone ID, call type ID, opposite mobile phone ID, start time, call duration, adversary ID, and call moment. CDL records cell (base station) ID and cell location. Base on CDR and CDL, we can get the personal communication data (Definition 1) in the base stations [20].

We elaborate our methods by the following two data sets.

Data set 1 is CDR data collected from January 1^{st}, 2008 to December 31^{st}, 2008.

Data set 2 is collected from an empirical experiment with 310 persons in the city we conducted from March 28^{th}, 2009 to March 31^{st}, 2009, including the CDR and real population situation (e.g., occupations and daily routes). Based on the domain knowledge, we classify these 310 persons into different categories and the persons in the same category have the similar communication patterns. We take this knowledge as ground truth.

Experiment environment: A server with four Intel Core Quad CPUs, Q9550 2.83 GHz and 32 GB main memory. In our experiment, we set one thread as a base station. The number of threads are the same as the number of base stations in our real life data set.

Comparison methods: We compare our work with (i) naïve method (as we discussed in Approach 1, Section III-C, ships all the distributed data to data center and conducts pattern matching) and (ii) Bloom Filter based method [7], [8].

B. Convergence study

In WBF, there are three parameters. The first one is a, the number of patterns as the input. The second one is b, the number of sample values, and the third one is k, the number of hashing functions. a and k are given by users, hence the setting of b should be studied in our experiment. We utilize four groups of data (four days data in *Data set 1*) to illustrate the impact of b on the pattern matching accuracy. The conclusion is that when the number of sample values (uniform sampling) is 5, the accuracy rates in different groups of data become converged, and when the number of sample values is 12, the accuracy rates in different groups of data become stable. Hence, in the following experiments, we set b as 12.

C. Accuracy and Efficiency Evaluation

We evaluate the accuracy and efficiency in four aspects, that is, precision, time cost, communication cost, and storage cost.

First, we introduce the definitions of evaluation metrics. Accuracy is defined by the precision, that is *True positive/(True positive+False positive)*; it is the fraction of retrieved similar patterns that are relevant to the search. Time cost is the time that our method consumes in searching the target patterns. Communication cost is the message size cost from pattern matching between base stations and data center. Storage cost is the space cost to store the patterns in base stations. The experiment data set is *Data set 1*. We evaluate three methods, naïve method (Naïve, ships data to data center and conducts pattern matching), Bloom Filter (BF, utilize a Bloom Filter in DI-matching, instead of WBF) and our Weighted Bloom Filter (WBF, utilize a Weighted Bloom Filter in DI-matching).

Second, we show and analyze the experiment results. In Figure 4 (a), x-axis is the number of patterns, and y-axis is the precision. The result shows that our method, WBF has the similar precision to the naïve method, which possesses the best precision because the naïve method conducts pattern matching after ships all the distributed data to data center, that is this method conducts pattern matching in the global collected data. At the same time, the BF has much less precision, especially as the number of patterns increases, because BF takes many different time series patterns as the similar ones while WBF can achieve much higher precision due to distinguishing the different time series patterns. In Figure 4 (b), x-axis is the number of patterns, and y-axis is the time cost. Our method is against the exponentially increasing time cost of the naïve method and the linearly increasing in BF, almost not sensitive to the number of patterns. This merit has been explained in the algorithm time complexity analysis. In Figure 4 (c), x-axis is the number of patterns, and y-axis is communication cost(in percentage of the naïve method). WBF performs much better than the other two methods. Because in a Weighted Bloom Filter, the matching number is greatly cut down by the weight-based scheme. In Figure 4 (d), x-axis is the number of patterns, and y-axis is the storage cost(in percentage of the naïve method). Even though we hash weights into the WBF, actually it

Table II
INCOMPLETE PATTERN MATCHING EFFECTIVENESS

Days	Precision	Recall	F1
March 28^{th}, 2009	0.98	0.99	0.98
March 29^{th}, 2009	0.99	0.99	0.99
March 30^{th}, 2009	0.97	0.99	0.98
March 31^{st}, 2009	0.98	0.99	0.98

brings little storage cost while as a trade-of, it achieves low communication cost and high accuracy.

To summarize, WBF can achieve high accuracy, low computation, storage, and communication cost. With the increasing number of patterns, WBF-based method saves more.

D. Effectiveness Evaluation

In Table II, we evaluate the incomplete pattern matching effectiveness on *Data set 2*. We take 310 persons as our study participants, among them there are six categories as shown in Figure 1. We take these persons and categories (similar patterns) as given patterns into our data set to test the effectiveness of the method. If the retrieved pattern is indeed a similar one based on the ground truth, we call it true, otherwise, false. In the evaluations, the precision is the fraction of retrieved similar patterns that are relevant to the search. The recall is defined as *True positive/(True positive+False negative)*; it is the fraction of the relevant similar patterns are successfully retrieved. F1 is the f-measure defined as *2(Precision*Recall)/(Precision+Recall)*. As shown in Table II, our methods perform well in empirical experiments, achieving more than 97% precision and 99% recall.

VI. RELATED WORK

Our work is mainly relative to two categories of current research work. The first category is the pattern matching method, and the second category is the distributed data mining. We review the two category work in terms of basic idea, the advantages and disadvantages as follows.

Pattern matching: Faloutsos et. al [13] studied a fast subsequence matching in time-series by the indexing. Bagnall et. al [6] proposed the clipping as a transformation for the shape similarity analysis of time series. Ahmed et. al [3] defined distributed pattern matching problem and presented a P2P-architecture solution. Agrawal et. al [2] presented the pattern matching over event streams in RFID-based inventory management. Agrawal et. al [1] studied the frequent pattern mining. Yang et. al [17] presented a shared execution strategy for processing multiple pattern mining requests over streaming data. Cole et. al [26] proposed a fast algorithm to conduct the approximate string matching, and Navarro et. al [25] summarized the techniques for approximate string matching in [5]. Our work is very different from the approximate string matching problem because approximate string matching is trying to conduct the string matching tolerant of errors, but our problem requires an accurate matching. At the same time, the distributed matching also makes the techniques for handling the approximate string matching not suitable, in terms of efficiency, scalability and accuracy. Chazelle et. al [23] generalized classical Bloom Filter with arbitrary functions as Bloomier Filter, while our WBF can handle the distributed and incomplete pattern matching. We are the first to propose *Incomplete Pattern Matching*, and study it in a distributed mobile environment.

Distributed data management: Zhao et. al [19] tried to retrieve items whose frequency of occurrence above a certain threshold in the distributed data sets. Our work is different from theirs in two aspects. First, their problem was to retrieve the frequent items, and the errors may come from the threshold, while in our problem, there is no such criteria, and we try to conduct the pattern matching. Second, they focused on the frequent patterns (items) mining in the static scenario, while our problem focuses on a dynamic mobile environment. Chen et. al [29] proposed the Peer-to-Peer multi-keyword searching utilizing an optimized Bloom Filter to cut a large amount of traffic cost. The similar works utilizing Bloom Filter to solve the distributed problems are also proposed in [27], [28], [30]. But in our new problem, we require high accuracy and scalability which can not be provided by the previous Bloom Filters. Babcock et. al [5] studied the top-k monitoring queries in distributed data streams. Zeinalipour-Yazti et. al [18] searched the distributed spatiotemporal similarity, that is, given a query trajectory Q, finding trajectories that follow a motion similar to Q, when each of the target trajectories is segmented across a number of distributed nodes. Approximate continuous querying over distributed streams was studied by Cormode et. al in [9]. Akdere et. al [4] proposed a plan-based complex event detection method across distributed sources. We are the first to study dynamic incomplete pattern matching in terms of high efficiency and effectiveness.

VII. CONCLUSION AND FUTURE WORK

In this paper, we present a new problem, *Incomplete Pattern Matching*, which considers pattern matching in a distributed mobile environment with incomplete data sets. To address the challenges from the incomplete data sets, we devise a novel *Weighted Bloom Filter*, and propose a framework, *DI-matching*, to efficiently and accurately retrieve the similar patterns in distributed incomplete data sets. Weighted Bloom Filter's properties are also analyzed. The empirical experiments confirm the merits of our methods.

In the future work, we will study more application scenarios, e.g., more distance functions for similarity computation. We also hope this paper can draw more attentions into the *Incomplete Pattern Matching* problem and our proposed method, *Weighted Bloom Filter*.

VIII. ACKNOWLEDGMENT

This research was supported in part by Hong Kong RGC Grant HKUST617710, China NSFC Grants 60933011, and National High Technology Research and Development (863) Program of China under Grant No. 2011AA010500. Siyuan Liu's research is in part supported by the Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office. Lei Chen's research is supported by National Grand Fundamental Research 973 Program of China under Project No. 2012CB316200, HP IRP Project 2011, Microsoft Research Asia Grant, MRA11EG05, HKUST SSRI11EG01, and NSFC No.60003074.

REFERENCES

[1] C. C. Aggarwal, Y. Li, J. Wang, and J. Wang. Frequent pattern mining with uncertain data. In *Proc. of the* 15^{th} *ACM SIGKDD*, 2009.

[2] J. Agrawal, Y. Diao, D. Gyllstrom, and N. Immerman. Efficient pattern matching over event streams. In *Proc. of the* 34^{th} *ACM SIGMOD*, 2008.

[3] R. Ahmed and R. Boutaba. Distributed pattern matching: A key to flexible and efficient p2p search. *IEEE Journal on Selected Areas in Communications*, 25(1), 2007.

[4] M. Akdere, U. Çetintemel, and N. Tatbul. Plan-based complex event detection across distributed sources. *Proc. VLDB Endow.*, 1(1), 2008.

[5] B. Babcock and C. Olston. Distributed top-k monitoring. In *Proc. of the* 29^{th} *ACM SIGMOD*, 2003.

[6] A. Bagnall, C. A. Ratanamahatana, E. Keogh, S. Lonardi, and G. Janacek. A bit level representation for time series data mining with shape based similarity. *Data Min. Knowl. Discov.*, 13(1), 2006.

[7] B. H. Bloom. Space/time trade-offs in hash coding with allowable errors. *Commun. ACM*, 13(7), 1970.

[8] A. Broder, M. Mitzenmacher, and A. Mitzenmacher. Network applications of bloom filters: A survey. In *Internet Mathematics*, 2002.

[9] G. Cormode and M. Garofalakis. Approximate continuous querying over distributed streams. *ACM Trans. Database Syst.*, 33(2), 2008.

[10] N. Du, C. Faloutsos, B. Wang, and L. Akoglu. Large human communication networks: patterns and a utility-driven generator. In *Proc. of the* 15^{th} *ACM SIGKDD*, 2009.

[11] N. Eagle and A. Pentland. Reality mining: sensing complex social systems. *Personal Ubiquitous Comput.*, 10(4), 2006.

[12] N. Eagle, A. Pentland, and D. Lazerc. Inferring social network structure using mobile phone data. *Proceedings of the National Academy of Sciences*, 106(36), 2009.

[13] C. Faloutsos, M. Ranganathan, and Y. Manolopoulos. Fast subsequence matching in time-series databases. In *Proc. of the* 20^{th} *ACM SIGMOD*, 1994.

[14] K. Farrahi and D. Gatica-Perez. What did you do today?: discovering daily routines from large-scale mobile data. In *Proc. of the* 16^{th} *ACM MM*, 2008.

[15] R. D. Malmgren, J. M. Hofman, L. A. Amaral, and D. J. Watts. Characterizing individual communication patterns. In *Proc. of the* 15^{th} *ACM SIGKDD*, 2009.

[16] H. Tong, C. Faloutsos, B. Gallagher, and T. Eliassi-Rad. Fast best-effort pattern matching in large attributed graphs. In *Proc. of the* 13^{rd} *ACM SIGKDD*, 2007.

[17] D. Yang, E. A. Rundensteiner, and M. O. Ward. A shared execution strategy for multiple pattern mining requests over streaming data. *Proc. VLDB Endow.*, 2(1), 2009.

[18] D. Zeinalipour-Yazti, S. Lin, and D. Gunopulos. Distributed spatio-temporal similarity search. In *Proc. of the* 15^{th} *ACM CIKM*, 2006.

[19] Q. G. Zhao, M. Ogihara, H. Wang, and J. J. Xu. Finding global icebergs over distributed data sets. In *Proc. of the* 25^{th} *ACM PODS*, 2006.

[20] Due to the anonymity of the review process, the website and the datasets will be published after the paper is accepted.

[21] S. Cohen and Y. Matias. Spectral bloom filters. In *Proc. of the* 29^{th} *ACM SIGMOD, 2003*.

[22] D. Guo, J. Wu, H. Chen, and X. Luo. Theory and network applications of dynamic bloom filters. In *Proc. of the* 25^{th} *IEEE INFOCOM, 2006*.

[23] B. Chazelle, J. Kilian, R. Rubinfeld, and A. Tal. The Bloomier filter: an efficient data structure for static support lookup tables. In *Proc. of the* 15^{th} *ACM-SIAM SODA, 2004*.

[24] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. *ACM Commun.*, 51(1), 2008.

[25] G. Navarro. A guided tour to approximate string matching. *ACM Comput. Surv.*, 33(1), 2001.

[26] R. Cole and R. Hariharan. Approximate string matching: a simpler faster algorithm. In *Proc. of the* 9^{th} *ACM-SIAM SODA, 1998*.

[27] H. Cai, P. Ge and J. Wang. Theory and Network Applications of Dynamic Bloom Filters. In *Proc. of the* 4^{th} *IEEE NAS, 2008*.

[28] D. Guo, J. Wu, H. Chen and X. Luo. Applications of Bloom Filters in Peer-to-peer Systems: Issues and Questions. In *Proc. of the* 25^{th} *IEEE INFOCOM, 2006*.

[29] H. Chen, H. Jin, L. Chen, Y. Liu and L. Ni. Optimizing Bloom Filter Settings in Peer-to-Peer Multi-keyword Searching. *IEEE TKDE*, 2011.

[30] F. Cuenca-acuna, C. Peery , R. Martin and T. Nguyen. PlanetP: Using Gossiping to Build Content Addressable Peer-to-Peer Information Sharing Communities. *IEEE Press*, 2003.